



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/626,251	07/23/2003	Julian Burger	3382-65690	1020

26119 7590 11/22/2006
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

EXAMINER

TECKLU, ISAAC TUKU

ART UNIT PAPER NUMBER

2192

DATE MAILED: 11/22/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

10/626,251

Applicant(s)

BURGER ET AL.

Examiner

Isaac T. Tecklu

Art Unit

2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 23 July 2003.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-44 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-44 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date See Continuation Sheet.
- 4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____.
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: _____.

Continuation of Attachment(s) 3). Information Disclosure Statement(s) (PTO/SB/08), Paper No(s)/Mail Date :07/10/06, 05/05/06, 12/23/05, 10/14/05, 07/30/04, 07/23/03.

DETAILED ACTION

1. This action is responsive to the application filed on 07/23/2003.
2. Claims 1-44 have been examined.

Oath/Declaration

3. The office acknowledges receipt of a properly signed oath/declaration filed on 07/23/2003.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claim 3 is rejected under 35 U.S.C 101 because the claimed invention is directed to non-statutory subject matter.
6. Claim 3 recites, "Computer data signal embodied in a carrier wave ". Thus, under the Interim Guidelines such wave does not fall within one of the four statutory classes of 35 U.S.C. 101 (See Annex IV). Therefore, the above claims are non-statutory.

A computer-readable media is a tangible physical article or object, some form of matter, which a signal /carrier wave is not. That the other two product classes, machine and composition of matter, require physical matter is evidence that a manufacture was also intended to require physical matter. A signal/carrier wave, a form of energy, does not fall within either of the two definitions of manufacture. Thus, a signal/carrier wave does not fall within one of the four statutory classes of Sec. 101.

Art Unit: 2192

See Annex IV (c) Electro-Magnetic Signals, Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility (signed October 26, 2005) – OG Cite: 1300 OG 142. Online version can be retrieved at

<http://www.uspto.gov/web/offices/com/sol/og/2005/week47/patgupa.htm>

Under the principles of compact prosecution, claim 3 has been examined as the Examiner anticipates the claims will be amended to obviate these 35 USC 101 issues. For example, A computer-readable physical storage medium...-

7. Claims 35-43 in particular, independent claims 35 and 41 are amount to non-statutory subject matter because the claims merely recite a data structure and/or program listing per se, thus they are non-functional descriptive materials. As such, the claims do not provide any action or interaction between the recited structural elements in order to enable a reasonable interpretation that a concrete, tangible, and useful result is present or yielded based on such interaction or actions taken.

Under the Interim Guidelines Section IV (b) non functional descriptive elements that do not constitute a statutory process, machine, manufacture or composition of matter are non statutory.

Claims 36-40 and 42-43 are also rejected for failing to cure the deficiencies of the above rejected non-statutory claim 35 and 41 respectively above.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Art Unit: 2192

9. Claims 1-44 are rejected under 35 U.S.C. 102(b) as being anticipated by Raverdy et al. (US 6,330,717 B1).

Per claim 1, Raverdy discloses a method of generating an extended version of software written in an object-oriented programming language which provides for object classes via a plurality of extensions to the software (e.g. FIGURE 5 and related text), the method comprising:

- receiving invocations of a plurality of software development scenario class extension sets comprising extensions for respective software development scenarios to be implemented by the extended version of the software (e.g. FIGURE 3, elements 320a-320c and related text); and
- extending one or more classes of the software as indicated by the extensions (col. 8: 15-20 "... automatically extends a constructor of the base class ...").

Per claim 2, this is the computer-readable media version of the claimed method discussed above (Claim 1), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Raverdy.

Per claim 3, this is the computer data signal version of the claimed method discussed above (Claim 1), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Raverdy.

Per claim 4, Raverdy discloses the method of claim 1 wherein the receiving for at least one of the extension sets occurs at runtime of the software (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 5, Raverdy discloses the method of claim 1 wherein the extending comprises outputting source code for the extensions (e.g. FIGURE 5, element 540 and related text).

Per claim 6, Raverdy discloses the method of claim 1 wherein the extension sets comprise: an extension set for implementing a target architecture (col. 8: 15-20 "... automatically extends a constructor of the base class ...").

Per claim 7, Raverdy discloses the method of claim 6 wherein the extension sets comprise: an extension set for implementing a compilation scenario (e.g. FIGURE 3, elements 320a-320c and related text).

Per claim 8, Raverdy discloses the method of claim 6 wherein the extension sets comprise an extension set for implementing a managed code scenario, and the method further comprises: based on the receiving, extending the classes to provide managed code functionality (col. 8: 15-20 "... extends a constructor of the base class ...").

Per claim 9, Raverdy discloses the method of claim 1 wherein an invocation for at least one of the extension sets indicates is received at runtime (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 10, Raverdy discloses the method of claim 1 wherein at least one of the extensions indicates an additional class member for at least one of the object classes of the software (e.g. FIGURE 7 and related text).

Per claim 11, Raverdy discloses a method of extending software written in a programming language by adding extensions to a core version of the software to generate an extended version of the software, the method comprising:

receiving a configuration of the extended version of software (e.g. FIGURE 5, element 510 and related text);

receiving in an object description language definitions of extensions to classes of the core version of the software according to the configuration of the extended version of the software, wherein the classes of the core version of the software are indicated in the object description language as statically extensible prior to compile time or dynamically extensible at runtime (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ..."); and

processing the classes of the core version of the software and the extensions to generate the extended version of the software (col. 8: 15-20 "... automatically extends a constructor of the base class ...").

Per claim 12, Raverdy discloses the method of claim 11, wherein the classes of the core version of the software indicated as being statically extensible are processed together with their corresponding extensions (col. 10: 25-30 "... generated on system statistics ...").

Per claim 13, Raverdy discloses the method of claim 12, wherein processing the classes of the core version of the software together with their corresponding extensions comprises:

using an object description language pre-processor to generate a header file with a source code representation of an extended class comprising the classes of the core version of the software and their corresponding extensions (e.g. FIGURE 5, element 510 and related text); and

compiling the header file to generate the extended version of the software (e.g. FIGURE 5, element 580 and related text).

Per claim 14, Raverdy discloses the method of claim 11, wherein the classes of the core version of the software indicated as being dynamically extensible are processed separate from their corresponding extensions (col. 8: 15-20 "... automatically extends a constructor of the base class ...").

Per claim 15, Raverdy discloses the method of claim 14, wherein processing the classes of the core version of the software separate from their corresponding extensions comprises:

using an object description language pre-processor for generating a header file comprising a source code version of the classes of the core version of the software (e.g. FIGURE 5, element 510 and related text); and

compiling the header file to generate a computer executable version of the classes of the core version of the software (e.g. FIGURE 5, element 580 and related text).

Per claim 16, Raverdy discloses the method of claim 14, wherein processing the classes of the core version of the software separate from their corresponding extensions comprises:

- using an object description language pre-processor for generating a header file comprising a source code version of the extensions to classes of the core version of the software (e.g. FIGURE 5, element 510 and related text); and
- compiling the header file to generate a computer executable version of the extensions to classes of the core version of the software (e.g. FIGURE 5, element 580 and related text).

Per claim 17, Raverdy discloses the method of claim 16, further comprising processing the computer executable version of the extensions to classes of the core version of the software to generate extended classes by linking the classes of the core version of the software to their respective extensions at runtime (e.g. FIGURE 5, element 570 and related text).

Per claim 18, Raverdy discloses the method of claim 11, wherein the definition of the classes of the core version of the software comprises one or more extension points for indicating points within code related to the core version of the software where code related to the extensions to classes of the core version of the software is injected (col. 25: 1-10 "... pre-defined instruction ...").

Per claim 19, Raverdy discloses the method of claim 11, wherein the core version of the software is an extensible core compiler framework and the extended version of the software is a customized compiler and receiving the configuration of the extended version of the software comprises obtaining a compiler type (e.g. FIGURE 5, element 521 and related text).

Per claim 20, Raverdy discloses the method of claim 19, wherein the compiler type is selected from a group consisting of a JIT compiler (e.g. FIGURE 5, element 520 and related text), a Pre-JIT compiler (col. 24: 55-60 "... pre-compiler ...") and a Native Optimizing Compiler (e.g. FIGURE 5, element 580 and related text).

Per claim 21, Raverdy discloses the method of claim 11, wherein the core version of the software is an extensible core software development tool framework and the extended version

of software is a customized software development tool and receiving the configuration of the extended version of software comprises obtaining a target type (e.g. FIGURE 5, element 510 and related text).

Per claim 22, Raverdy discloses the method of claim 11, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and receiving the configuration of the extended version of software comprises obtaining a feature type (e.g. FIGURE 5, element 580 and related text).

Per claim 23, Raverdy discloses a system for extending software by adding extensions to a core version of the software to generate an extended version of the software, the system comprising: an object description language pre-processor operable for receiving extensions to classes of the core version of the software in an object description language and generating a source code version of the extensions to classes of the core version of the software (e.g. FIGURE 3, elements 320a-320c and related text), wherein the classes of the core version of the software are indicated in the object description language as being dynamically extensible at runtime or statically extensible prior to compile time (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 24, Raverdy discloses the system of claim 23, wherein the object description language pre-processor is operable for processing the classes of the core version of the software indicated as being statically extensible together with their corresponding extensions to generate a source code version of extended classes comprising the source code version of the classes of the core version of the software and their corresponding extensions (col. 8: 15-20 "... extends a constructor of the base class ...").

Per claim 25, Raverdy discloses the system of claim 24, further comprising a compiler for compiling the source code version of the extended classes to generate a computer-executable

version of the extended classes to be used for generating the extended version of the software (e.g. FIGURE 5, element 560 and related text).

Per claim 26, Raverdy discloses the system of claim 23, wherein the classes of the core version of the software further comprise extension points for indicating locations within code related to the core version of the software where code related to the extensions are injected (col. 25: 1-10 "... pre-defined instruction ...").

Per claim 27, Raverdy discloses the system of claim 23, wherein the object description language pre-processor is programmed for processing the classes of the core version of the software indicated as being dynamically extensible separate from their corresponding extensions (e.g. FIGURE 3, elements 320a-320c and related text).

Per claim 28, Raverdy discloses the system of claim 27, further comprising a compiler for compiling the source code version of the extensions to the classes of the core version of the software to generate a computer-executable version of the extensions (e.g. FIGURE 5, element 570 and related text).

Per claim 29, Raverdy discloses the system of claim 28, further comprising a computer processor for executing the computer-executable version of the extensions to generate an extended version of the software at runtime by linking the classes of the core version of the software to their corresponding extensions (col. 8: 15-20 "... extends a constructor of the base class ...").

Per claim 30, Raverdy discloses the system of claim 29, wherein the processor executes the computer-executable version of the extensions by invoking a computer-executable version of the core version of the software and injecting the extensions into code related to the core version of the software at runtime (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 31, Raverdy discloses the system of claim 23, wherein the extensions correspond to a configuration of the extended version of the software (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 32, Raverdy discloses the system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a target type (e.g. FIGURE 7 and related text).

Per claim 33, Raverdy discloses the system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a compiler type (e.g. FIGURE 5, element 560 and related text).

Per claim 34, Raverdy discloses the system of claim 31, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises a feature type (e.g. FIGURE 5, element 570 and related text).

Per claim 35, Raverdy discloses a computer readable storage medium having stored thereon class declarations of classes of a core version of a software to be extended by extension declarations for extending the core version of the software to generate an extended version of the software, the class declaration of the core version of the software (col. 8: 15-20 "... automatically extends a constructor of the base class ...") comprising:

- one or more core class members (col. 7: 1-5 "...object for those classes ...");
- and one or more extensibility attributes of the core classes, wherein the extensibility attribute indicates that the core classes are either statically extensible prior to compile time or dynamically extensible at runtime (col. 10: 25-30 "... generated on system statistics ...").

Per claim 36, Raverdy discloses the computer readable storage medium of claim 35, further comprising the extension declarations, wherein the extension declarations comprise one

Art Unit: 2192

or more extension class members for extending the core version of the software (e.g. FIGURE 5, element 520 and related text).

Per claim 37, Raverdy discloses the computer readable storage medium of claim 36, wherein the extension declarations correspond to a particular configuration of the extended version of the software and the extension declaration further comprise one or more attribute declarations for indicating the configuration of the extended version of the software (col. 10: 25-30 "... generated on system statistics ...").

Per claim 38, Raverdy discloses the computer readable storage medium of claim 37, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and choosing the configuration of the extended version of software comprises choosing a compiler type (e.g. FIGURE 5, element 520 and related text).

Per claim 39, Raverdy discloses the computer readable storage medium of claim 37, wherein the core version of the software is an extensible core software development tool framework and the extended version of software is a customized software development tool and the configuration of the extended version of software comprises choosing a target type (e.g. FIGURE 5, element 530 and related text).

Per claim 40, Raverdy discloses the computer readable storage medium of claim 35, further comprising extension points for indicating locations within code related to the core version of the software where code related to their corresponding extensions should be injected (col. 8: 20-25 "... extended so that the new object will register itself to the run-time ...").

Per claim 41, this is the computer-readable storage medium version of the claimed system discussed above (Claim 23), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Raverdy.

Per claim 42, this is the computer-readable storage medium version of the claimed system discussed above (Claim 24), wherein all claim limitations have been addressed and/or covered in cited areas as set forth above. Thus, accordingly, these claims are also anticipated by Raverdy.

Per claim 43, Raverdy discloses the computer readable medium of claim 41, wherein the pre-processor program is further operable for using the extensions of the classes of the core version of the software received in form of the object description language to generate a source code version of the extensions to be used for linking the extensions to their corresponding classes of the core version of the software at runtime to extend the core version of the software (e.g. FIGURE 5, element 570 and related text).

Per claim 44, Raverdy discloses a system for extending software by adding extensions to a core version of the software to generate an extended version of the software, the system comprising:

means for receiving extensions to classes of the core version of the software in an object description language and generating a source code version of the extensions to classes of the core version of the software, wherein the classes of the core version of the software are indicated in the object description language as being dynamically extensible at runtime or statically extensible prior to compile time (e.g. FIGURE 3, elements 320a-320c and related text); and

means for compiling the source code version of the extensions to classes of the core version of the software to be used for generating the extended version of the software (col. 8: 15-20 "... automatically extends a constructor of the base class ...").

Art Unit: 2192


Conclusion

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Isaac T. Tecklu whose telephone number is (571) 272-7957. The examiner can normally be reached on M-TH 9:300A - 8:00P.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Isaac Tecklu
Art Unit 2192


TUAN DAM
SUPERVISORY PATENT EXAMINER